CONTROL DATA® 6000 SERIES COMPUTER SYSTEMS

FORTRAN Extended General Information Manual

Additional copies of this manual may be obtained from the nearest Control Data Corporation Sales office listed on the back cover.

October, 1966
Pub. No. 60176400

CONTROL DATA CORPORATION
Documentation Department
3145 PORTER DRIVE
PALO ALTO, CALIFORNIA

©1966, Control Data Corporation
Printed in the United States of America

INTRODUCTION

FORTRAN for the CONTROL DATA ® 6400/6600 computer system is a procedural language designed for solving problems of a mathematical or scientific nature. The source language is fully compatible with ASA FORTRAN. Several additional features are included in the language. These serve to increase the power of FORTRAN as a solution tool and broaden the scope of the language so that many existing FORTRAN systems become acceptable subsets of 6400/6600 FORTRAN.

The compiler is designed to produce object code which takes full advantage of the high speed execution characteristics of the 6400/6600 computer systems.

LANGUAGE FEATURES

- Constants and variables of several types:
  - integer
  - single precision floating point (real)
  - double precision floating point
  - complex
  - octal
  - hollerith
  - logical
- Mixed mode arithmetic
- Masking (Boolean), logical and relational operators
- Shorthand notation for logical operators and constants
- Library functions
- Independently compilable subprograms
- Multiple entry points to subroutines and functions
- Multiple subroutine exits
- Expressions as subscripts
- Variable dimensions
- Variable FORMAT capability
- Conversion formats for all data forms
- Specification and I/O statements to allow use of ECS
- Array reference with fewer subscripts than dimensioned
- Hollerith constants in expressions and Data statements
- More than one statement per line

- Left or right-justified hollerith constants
- Two-branch IF statements
- NAMELIST capability

CODE OPTIMIZATION

Efficient code is the primary design objective of the 6400/ 6600 FORTRAN compiler. Extensions to the optimization capability of previous compilers include:

- Elimination of redundant operations where possible.
- Evaluation of array element address by the index function method.
- Critical path analysis of instruction sequences to maximize parallel operation.
- Reformation of subexpressions to permit extended parallelism.
- Evaluation of constant subexpressions at compile time.
- Determination at compile time for each reference to a formal parameter to determine whether it should be referred to by address substitution or indirect addressing.
- Elimination of common remote parameter lists.
- Formation of simple constants by sets rather than loads.
- Elimination of branches to the next instruction.
- Presetting of arrays with a constant pattern is specially handled at load time to avoid the generation of a large binary deck.
- Inline evaluation of some functions.

The FORTRAN compiler processes each subprogram independently using a two-pass technique. The source language is read once from the input device. The output consists of object code, COMPASS listings and a source listing with diagnostic messages.

FORTRAN operates under the SCOPE operating system. The objects code produced by the compiler is designed to operate under SCOPE.

LANGUAGE ELEMENTS 1

CONSTANTS

The following kinds of constants are allowed:

INTEGER

18 decimal digits or less with a range in magnitude of 0 through $2^{59}-1$

FLOATING POINT

real constants:	14 decimal digits or less; magnitude range of 10^{-294} through 10^{322} and 0
double constants:	28 decimal digits or less; magnitude range of 10^{-294} through 10^{322} and 0
complex constants:	two real constants.

Two machine words are used to store double and complex constants.

OCTAL

Octal constants of up to 20 digits may be defined directly in the FORTRAN program as well as entered by input or data statements.

HOLLERITH

1 to 136 alphanumeric or special characters may be given as a constant left justified, blank filled.

1 to 10 characters may be right justified with zero fill. Hollerith constants, also, may be defined directly in the program.

LOGICAL

The symbolic constants that represent the logical values true and false are:

.TRUE.	.T.
.FALSE.	.F.

VARIABLES

Variables may be simple or subscripted, and a subscript may contain up to three subscripts.

REPLACEMENT STATEMENTS AND EXPRESSIONS

REPLACEMENT STATEMENT

The general form of the replacement statement is:

R = E

where R is a variable and E is an expression. Expressions may be arithmetic, masking, relational or logical.

The replacement statement is the only FORTRAN statement that does not rely on a verb or declarator to describe its action.

Here, the character " = " is defined to mean "is replaced by".

For example the replacement statement

A = B

can be read as: the value of variable A is replaced by the value of the variable B.

The statement

I = J + (K**2)/5

instructs the processor to evaluate the expression on the right by squaring the variable K, dividing the result by the constant 5, adding the quotient to the variable J and assigning that sum to the variable I.

ARITHMETIC EXPRESSIONS

Arithmetic operators are:

**	exponentiation
/	division
*	multiplication
+	addition
-	subtraction

Mixed mode expressions are allowed; any type (except logical) of variable or constant may be combined with any other type of variable or constant.

MASKING EXPRESSIONS

Masking operators are:

.AND.	logical product
.OR.	logical sum
.NOT.	complement

The statement:

```
PROD = ABLE .AND. BAKER
```

directs the processor to form the logical product of the variables ABLE and BAKER and assign the result to the variable PROD. In the general form of the masking expressions, p op v, the operations are as follows:

p	v	p.AND.v	p.OR.v	.NOT.p
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

RELATIONAL EXPRESSIONS

Relational operators are:

.EQ.	equal to
.NE.	not equal to
.GT.	greater than
.GE.	greater than or equal to
.LT.	less than
.LE.	less than or equal to

The value of the expression, q_1 op q_2, is true if q_1 stands in the specified relation to q_2, otherwise it is false.

LOGICAL EXPRESSIONS

The logical operators are:

.AND.	conjunction
.OR.	disjunction
.NOT.	negation

The value of the logical expression, o_1 op o_2 . . .op o_n, is either true or false. The o_i are relational expressions or variables of type logical.

(NOTE: The logical and masking operators may be abbreviated with .A. for .AND., .O. for .OR., and .N. for .NOT.)

STATEMENT IDENTIFIERS

Statement identifiers provide numbers for reference to statements. A statement identifier may be 1 through 99999.

GO TO STATEMENTS

GO TO statements transfer control to a specified statement.

GO TO n

transfers control unconditionally to statement n.

GO TO i, $(n_1, n_2, \ldots n_m)$

transfers control to the statement identified by i, an integer variable, which has previously been assigned an integer value by the statement ASSIGN n_i to i. The parenthetical list of statement numbers is optional.

GO TO$(n_1, n_2, \ldots, n_m)$, e

transfers control to the statement identified by n_i, where i is the integer value, the arithmetic expression e.

IF STATEMENTS

IF statements transfer control conditionally, depending upon the value of an expression.

IF (expressions) n_1, n_2, n_3

transfers control to statement n_1 if the value of the arithmetic or masking expression is negative. If the value is zero, control is transferred to statement n_2. A positive value transfers control to n_3.

IF (logical expression)s
If the logical expression is true the imperative statement(s) is executed. Otherwise, control is transferred around it.

IF (expression) n_1, n_2

arithmetic expression: transfer to statement n_1 if non-zero, otherwise to statement n_2.

logical expression: transfer to statement n_1 if the expression is true, transfer to statement n_2 if false.

DO STATEMENTS

DO n i=m_1, m_2, m_3

Repeats the execution of all succeeding statements up to and including statement number n. The index, i, is an integer variable initially set to m_1. The DO loop is repeated the number of times necessary for i to attain the value of m_2, incremented each time through the loop by the value of m_3.

CONTINUE

CONTINUE

Provides a no-operation instruction which transfers control to the next instruction in sequence. It may be used to terminate a DO loop when the last statement in the loop would otherwise be a control transfer statement.

PAUSE

PAUSE

or

PAUSE n

Causes a cessation of program operation. Execution may be resumed via SCOPE.

STOP

STOP

or

STOP n

Causes program termination.

DATA SPECIFICATION STATEMENTS

DIMENSION

DIMENSION $v_1(s_1,s_2,s_3),v_2(s_1,s_2,s_3),\ldots$

Reserves memory locations for the arrays $v_1,v_2,\ldots$

TYPE

INTEGER list

REAL list

DOUBLE PRECISION list or DOUBLE list

COMPLEX list

LOGICAL list

Each of the above forms defines a type of FORTRAN variable. The list is a string of variable names. The names may be followed by dimension information to provide a further means of defining arrays.

COMMON

COMMON/name_1/list_1/name_2/$\text{list}_2\ldots$

Reserves memory locations in a named common block. All variables (or arrays) in the list are contained in the block designated by the corresponding name. One common block may be unnamed (blank). Common blocks may be assigned to Extended Core Storage (ECS) by prefixing the block name with an asterisk.

EQUIVALENCE

EQUIVALENCE $(v_1,v_2,v_3,\ldots),(v_5,v_6,v_7,\ldots),\ldots$

Assigns variables (simple or subscripted) enclosed within each set of parentheses to the same memory locations. Thus, EQUIVALENCE allows renaming of FORTRAN variables to suit the programmers convenience.

EXTERNAL

EXTERNAL $n_1, n_2, n_3, \ldots$

Defines the names (n_i) as external procedures.

DATA

DATA (i_1 = value list), (i_2 = value list), . . .

or

DATA $i_1, i_2, \ldots$/value list/$i_3, i_4, \ldots$/value list

Enables initialization of variables at load time. Each i is a variable name or an array element name. The constants in the value lists correspond to the identifiers and are stored into locations assigned to the indicated variables when the FORTRAN subprogram is loaded by SCOPE.

SUBPROGRAM STATEMENTS

PROGRAM

PROGRAM ($f_1, f_2, \ldots f_n$)

Defines the files (f_i) to be used by the subprogram.

END

END

Is the last statement of a subprogram.

FUNCTION

A function in FORTRAN terms is an arithmetic procedure which yields a single-value result. Functions may be declared in two ways:

The arithmetic statement function is a macro facility offered by the compiler. The programmer may, for example, write at the beginning of a subprogram:

```
FUN(X,A,B,C)=A*X**2+B*X+C
```

Then in the body of the subprogram he may write:

```
Z=(FUN(Y,5.,2.,4.))/2.0
```

and expect the function FUN to be performed at the appropriate time in the expression evaluation. An arithmetic statement function is valid only within the subprogram in which it is defined.

An independent function is a subprogram bounded by the statements:

FUNCTION f($p_1, p_2, \ldots p_n$)

.

.

.

END

f is the function name and p_i the formal parameters. An independent function is called in the same manner as an arithmetic statement function.

SUBROUTINE

SUBROUTINE $s(p_1,p_2,\ldots p_n)$RETURNS$(v_1,v_2,\ldots v_n)$

.
.
.

END

The above two statements are the delimiters for a subroutine subprogram. s is the subroutine name, and p_i the formal parameters.

A subprogram may include, as formal parameters, an array identifier and its dimensions in simple integer variable form. The actual dimensions are specified by the calling subprogram. The RETURNS phrase is optional. Each variable, v, is associated with a statement label specified in the CALL statement.

CALL

CALL $s(p_1,p_2,\ldots p_n)$RETURNS$(n_1,n_2,\ldots n_n)$

Transfers control to a subroutine subprogram; s is the program name, and p_i the actual parameters.

RETURNS is optional here, also. If omitted, control returns from the subroutine to the statement following the call. Otherwise each n is a statement label to which the called subroutine may return.

ENTRY

ENTRY name

Identifies alternate entry points to a function or subroutine. When a subprogram is called by one of these alternate names, execution begins with the indicated statement.

RETURN

RETURN v

Used within a function or subroutine to return control to the calling subprogram. An optional variable name (v) may be used to indicate a nonstandard return.

The general form of a data transmission statement is:

operation n, data list

The operation specifies the transmission process and unit; n refers to a FORMAT statement and the data list specifies the variables (storage locations) involved. In binary tape operations, no FORMAT statement is necessary.

DATA LIST

The data list consists of any number of simple or subscripted variables, separated by commas. If an array name appears without subscripts, the whole array is transmitted. Arrays may also be transmitted using notation similar to the DO loop notation:

$$(((A(I,J,K,)I=1_1, 1_2, 1_3), J=m_1, m_2, m_3)K=n_1, n_2, n_3)$$

FORMAT STATEMENT

$$\text{FORMAT } (s_1, s_2, \ldots k(s_a, s_b, \ldots) \ldots s_n)$$

Defines the structure of BCD data. s_i are the format specifications and k is a repetition factor:

$$\ldots, k_1(s_a, s_b, \ldots, k_2(\ldots, k_{10})s_x, s_y, \ldots))))))) \ldots$$

Specifications are repeated from the last open parenthesis until the list is exhausted. Format specifications may be any of the following:

Ew.d	Single or double precision floating point conversion depending upon variable type
Fw.d	Single precision floating point conversion without explicit exponent field
Gw.d	Combination of E and F formats depending on variable magnitude

T	Assigns beginning column for subsequent information
Iw	Decimal integer conversion
Ow	Octal integer conversion
Aw	Alphanumeric conversion left justified in storage with blank fill
Rw	Alphanumeric conversion, right justified in storage with zero fill
Lw	Logical conversion
wHf	Heading and labeling information
f	Heading and labeling information
wX	Intra-record spacing
/	Inter-record spacing

Format control may be variable. An array element name is used in an input/output statement in place of a FORMAT identifier.

PUNCHED CARD RECORDS

READ n,list

PUNCH n,list

Punched card records may be transmitted to the punch output unit or from the standard input unit.

PRINTER RECORDS

PRINT n,list

Transmits records to the standard output unit.

BCD RECORDS

READ (u,n)list

WRITE (u,n)list

Transmit records between memory and logical unit u.

BINARY RECORDS

READ (u)list

WRITE (u)list

Transmit binary records to or from the logical unit specified.

NON-STANDARD STATEMENTS

BUFFER IN (u,p)(fi,li)

BUFFER OUT (u,p)(fi,li)

u is a logical unit number

p is a parity key

fi is the identifier of the first word of the block to be transmitted

li is the identifier of the last word of the block to be transmitted

READ ECS (cmi,eci,n)

WRITE ECS (cmi,eci,n)

cmi is the identifier of a central memory address

eci is the identifier of an address in Extended Core Storage

n is the number of words to be transmitted

READ MS (fn(i),k)

WRITE MS (fn(i),k)

fn is a file identifier in mass storage (disk, drum, etc. . .)

i is a record ordinal within the file

k is a FORMAT statement number

NAMELIST

The NAMELIST method of BCD I/O offers a simple technique for processing data in a free format.

NAMELIST /n/v_1,v_2,v_3,...

Identifies the variables v_1,v_2,...etc. as belonging to the name list n.

READ (1,n)

Accepts input items which resemble replacement statements, 1 is a logical unit and n is the namelist name.

For example:

A=14.0,B=17.9,XTRA=0,B(3,1)=1.,2.,7.5

represent samples of NAMELIST input.

WRITE (1,n)

Produces output in the same form.

MAGNETIC TAPE

REWIND u

BACKSPACE u

END FILE u

If a REWIND or BACKSPACE is the next I/O operation after WRITE (tape) the FORTRAN I/O routine will write an end-of-file, backspace over it, and then execute the command. u is a logical unit number.

CONTROL DATA

CORPORATION

COMMENT AND EVALUATION SHEET

6000 Computer Systems

FORTRAN Extended General Information Manual

Pub. No. 60176400 October, 1966

THIS FORM IS NOT INTENDED TO BE USED AS AN ORDER BLANK. YOUR EVALUATION OF THIS MANUAL WILL BE WELCOMED BY CONTROL DATA CORPORATION. ANY ERRORS, SUGGESTED ADDITIONS OR DELETIONS, OR GENERAL COMMENTS MAY BE MADE BELOW. PLEASE INCLUDE PAGE NUMBER REFERENCE.

FROM NAME : ______________________________

BUSINESS ADDRESS : ______________________________

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Documentation Department
3145 PORTER DRIVE
PALO ALTO, CALIFORNIA

FOLD FOLD

STAPLE STAPLE

6000 FORTRAN Extended General Information Manual

CONTROL DATA SALES OFFICES

ALAMOGORDO, NEW MEXICO
ALBUQUERQUE, NEW MEXICO
ATLANTA, GEORGIA
AUSTIN, TEXAS
BILLINGS, MONTANA
BIRMINGHAM, ALABAMA
BOSTON, MASSACHUSETTS
BOULDER, COLORADO
CAPE CANAVERAL, FLORIDA
CEDAR RAPIDS, IOWA
CHICAGO, ILLINOIS
CINCINNATI, OHIO
CLEVELAND, OHIO
COLORADO SPRINGS, COLORADO
DALLAS, TEXAS
DAYTON, OHIO
DENVER, COLORADO
DETROIT, MICHIGAN
DOWNEY, CALIFORNIA
GREENSBORO, NORTH CAROLINA
HARTFORD, CONNECTICUT
HONOLULU, HAWAII
HOUSTON, TEXAS
HUNTSVILLE, ALABAMA
IDAHO FALLS, IDAHO
INDIANAPOLIS, INDIANA
KANSAS CITY, KANSAS
LAS VEGAS, NEVADA
LIVERMORE, CALIFORNIA
LOS ANGELES, CALIFORNIA
MADISON, WISCONSIN
MIAMI, FLORIDA
MILWAUKEE, WISCONSIN
MINNEAPOLIS, MINNESOTA
MONTEREY, CALIFORNIA
NEWARK, NEW JERSEY
NEW ORLEANS, LOUISIANA
NEW YORK, NEW YORK
OAKLAND, CALIFORNIA
OMAHA, NEBRASKA
PALO ALTO, CALIFORNIA
PHILADELPHIA, PENNSYLVANIA
PHOENIX, ARIZONA
PITTSBURGH, PENNSYLVANIA
PORTLAND, OREGON
ROCHESTER, NEW YORK
SACRAMENTO, CALIFORNIA
ST. LOUIS, MISSOURI
SALT LAKE CITY, UTAH
SAN BERNARDINO, CALIFORNIA
SAN DIEGO, CALIFORNIA
SAN FRANCISCO, CALIFORNIA
SAN JUAN, PUERTO RICO
SANTA BARBARA, CALIFORNIA
SEATTLE, WASHINGTON
TULSA, OKLAHOMA
VIRGINIA BEACH, VIRGINIA
WASHINGTON, D. C.

ADELAIDE, AUSTRALIA
AMERSFOORT, THE NETHERLANDS
AMSTERDAM, THE NETHERLANDS
ATHENS, GREECE
BOMBAY, INDIA
CALGARY, ALBERTA, CANADA
CANBERRA, AUSTRALIA
DUSSELDORF, GERMANY
FRANKFURT, GERMANY
GENEVA, SWITZERLAND
HAMBURG, GERMANY
JOHANNESBURG, SOUTH AFRICA
KASTRUP, DENMARK
LONDON, ENGLAND
LUCERNE, SWITZERLAND
MELBOURNE, AUSTRALIA
MEXICO CITY, MEXICO
MONTREAL, QUEBEC, CANADA
MUNICH, GERMANY
OSLO, NORWAY
OTTAWA, ONTARIO, CANADA
PARIS, FRANCE
ROME, ITALY
STOCKHOLM, SWEDEN
STUTTGART, GERMANY
SYDNEY, AUSTRALIA
TEHERAN, IRAN
TEL AVIV, ISRAEL
TOKYO, JAPAN (C. ITOH ELECTRONIC COMPUTING SERVICE CO. LTD.)
TORONTO, ONTARIO, CANADA
VANCOUVER, BRITISH COLUMBIA, CANADA
ZURICH, SWITZERLAND

8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440

CONTROL DATA
CORPORATION

Pub. No. 60176400

Litho in U.S.A.